PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-260826
(43)Date of publication of application : 29.09.1998

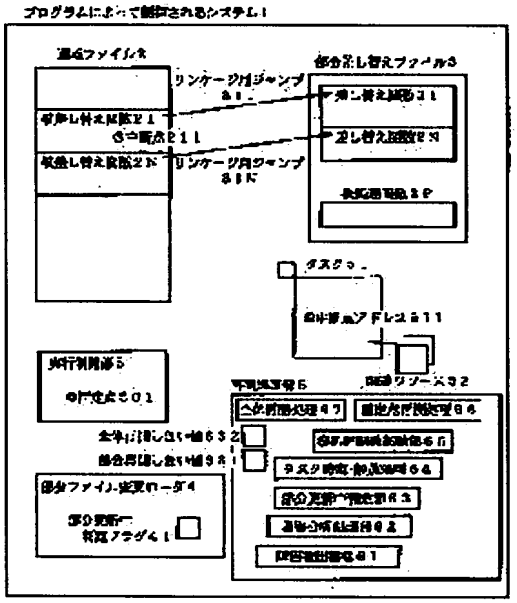
(51)Int.Cl. G06F 9/06
G06F 9/46
G06F 12/00
G06F 12/00
G06F 12/00

(21)Application number : 09-062754
(22)Date of filing : 17.03.1997

(71)Applicant : NIPPON TELEGR & TELEPH CORP <NTT>
NEC CORP
(72)Inventor : SUNAGA HIROSHI
NAKAMURA RYOICHI
YAMADA TETSUYASU
OCHI KENICHI

(54) PARTIAL FILE UPDATING METHOD DURING SYSTEM OPERATION

(57)Abstract:
PROBLEM TO BE SOLVED: To prevent the operation interruption of an entire system with receiving only a partial effect even though a conflict occurs because of an interrupting task and it might be a system failure at the time of partially changing an operation file and also to make the system perform normal operation recovery even when a fault is repeated in spite of excution of the main processing.
SOLUTION: A failure detecting mechanism 61 detects a failure, a failure analysis processing part 62 analyzes if the failure is caused by a software factor, a task specification and release processing part 64 specifies a program parallel execution processing unit that generates the failure when it is caused by the software factor and resets the task 51 in which a conflict occurs and at the same time, a post-processing function starting part 65 specifies a resource about 52 about the task and starts a function that initializes. A fixed point restart processing part 66 resumes the operation of a system at the fixed point 501 of an execution controlling part 5.



LEGAL STATUS

[Date of request for examination]	17.05.2001
[Date of sending the examiner's decision of rejection]	04.03.2005
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	
[Date of registration]	
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－260826

(43)公開日　平成10年(1998)９月29日

(51)Int.Cl.[6]		識別記号	FI		
Ｇ０６Ｆ	9/06	５４０	Ｇ０６Ｆ	9/06	５４０Ｆ
	9/46	３３０		9/46	３３０Ｃ
	12/00	５１８		12/00	５１８Ａ
		５３１			５３１Ｒ
		５３５			５３５Ｚ

審査請求　未請求　請求項の数４　ＯＬ　（全 10 頁）

(21)出願番号　特願平9－62754

(22)出願日　平成９年(1997)３月17日

(71)出願人　000004226
日本電信電話株式会社
東京都新宿区西新宿三丁目19番２号

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号

(72)発明者　須永　宏
東京都新宿区西新宿三丁目19番２号　日本電信電話株式会社内

(72)発明者　中村　亮一
東京都新宿区西新宿三丁目19番２号　日本電信電話株式会社内

(74)代理人　弁理士　磯村　雅俊　（外１名）

最終頁に続く

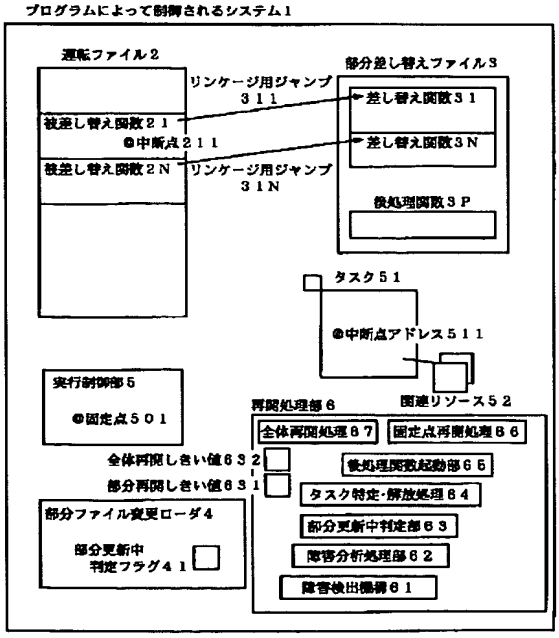
(54)【発明の名称】　システム運転中部分ファイル更新方法

(57)【要約】

【課題】運転ファイルを部分変更する際に、中断タスクのために矛盾が発生しシステム障害となるところ、部分的影響のみでシステム全体の運転中断を防止するとともに、本処理を行っても罹障を繰り返す場合でも、システムの正常な運転回復を行わせる。

【解決手段】障害検出機構６１で障害を検出し、障害分析処理部６２でその障害がソフトウェア要因によるものかを分析し、そうであれば、タスク特定・解放処理部６４で障害を発生させたプログラム並列実行処理単位を特定して、矛盾が発生したタスク５１をリセットすると同時に、後処理関数起動部６５がそのタスクに関するリソース５２を特定して初期設定する関数を起動する。固定点再開処理部６６は、実行制御部５の固定点５０１からシステムの運転を再開させる。

本発明の原理ブロック図



1

【特許請求の範囲】
【請求項１】運転ファイルに対して部分差し替えファイルをロード・リンクしてプログラムの変更を行う通信・情報処理システムのファイル更新方法において、
部分ファイル更新ローダが上記運転ファイルの被差し替え部分を新しい部分差し替えファイルに変更している際に、該部分差し替えファイルが組み込まれてから予め定めた監視時間内に発生した障害を検出し、
該障害がソフトウェア要因によるものか否かを分析し、分析の結果、ソフトウェア要因であれば、該障害を発生させたプログラム並列実行処理単位を特定して、特定された該実行単位単位を消去し、
プログラム実行制御の特定の点から処理を再開させることを特徴とするシステム運転中部分ファイル更新方法。
【請求項２】請求項１に記載のシステム運転中部分ファイル更新方法において、
前記部分ファイル更新ローダが運転ファイルの被差し替え部分を新しい部分差し替えファイルに変更する際に、該部分差し替えファイルを運転中のシステムに投入すると同時に、該部分差し替えファイルの盛り込みにより発生し得るソフトウェア障害に関するリソースを解放するために記述した後処理プログラムを該システムに割り付け、
運転中のファイルを部分的に差し替えている期間中に、ソフトウェア障害が発生した場合には、該ソフトウェア障害を引き起こしたプログラム並列実行処理単位を消去した後、上記後処理プログラムを起動させて該処理プログラムによるリソースを解放し、
該リソース解放処理後にプログラム実行制御の特定の点から処理を再開させることを特徴とするシステム運転中部分ファイル更新方法。
【請求項３】請求項２に記載のシステム運転中部分ファイル更新方法において、
前記運転中のファイルを部分的に差し替えている期間中に発生するソフトウェア要因の障害発生回数のしきい値を指定する手段を設け、
システムは上記期間中の障害発生回数を計数して、上記指定手段のしきい値に達した時点で該当する部分差し替えファイルを取り外すことを特徴とするシステム運転中部分ファイル更新方法。
【請求項４】請求項３に記載のシステム運転中部分ファイル更新方法において、
前記前記運転中のファイルを部分的に差し替えている期間中に発生するソフトウェア要因の障害発生回数について少なくとも２種類のしきい値を指定する手段を設け、
システムは上記期間中の障害発生回数を計数して、先ず最初のしきい値の回数に達した時点で、該当する部分差し替えファイルを取り外し、
該部分差し替えファイルの取り外し後、再度障害が発生して次のしきい値の回数に達した時点で、該システム全体を再開させることを特徴とするシステム運転中部分ファイル更新方法。

2

【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、プログラムにより運転の制御が行われるシステムのファイル管理方法に関し、特に運転ファイルを部分的に変更する際に、複数箇所の関数を修正しても中断タスクがあるために矛盾が発生してシステム障害になるところ、部分的な影響のみでシステム全体の運転の妨げをなくし、システム障害の可能性を低減することができるようなシステム運転中部分ファイル更新方法に関する。
【０００２】
【従来の技術】従来より、プログラムにより運転されるシステムでは、プログラムの瑕疵の修正、機能の追加等によりファイルを部分的に差し替えてプログラムの変更を行っている。この場合に、部分差し替えファイル投入直前でタスクが中断していると、部分差し替えファイルが割り付けられた直後にそのタスクが中断点から再開した場合、本タスクが格納しているスタック情報と部分差し替えファイルの新プログラム処理との間で処理矛盾が生じて、システム異常になる可能性がある。そのために、システム異常を回避しながら運転中ファイルの変更を行うことが重要となる。図１３は、従来のシステム構成例を示すブロック図である。図１３において、１はプログラムにより制御されるシステム、２はシステム建設時、あるいはファイル全体更新時に設定された原本の運転ファイル、２１～２Ｎは運転ファイルを構成するプログラム単位（ここでは、関数と呼ぶ）のうち、置換される対象のもの、２１１は被差し替え関数の中断点、３は部分差し替えファイル、３１，３Ｎは部分差し替えファイルを構成する差し替え対象の関数、４は差し替え関数をメモリ上に割り付けて、原本ファイルの被差し替え対象関数の代りにリンクし直す機能を備えた部分ファイル変更ローダ、４１は部分更新中判定フラグ、５は実行制御部、５０１は固定点、５１は実行制御部により提供されるタスクと呼ばれる並列実行単位、６は再開処理部、６１は障害検出部、６２は障害分析処理部、６７は全体再開処理部である。なお、上記障害検出部６１は、未実装番地やプロテクトのかかった命令エリアを走行したりすると、ハードウェア的に割り込みを起こさせるプロセッサ内のハードウェア機構であるが、実行制御部５、タスク５１、および再開処理部６のその他機能部は、メモリ上に格納され、事象が発生した際にプロセッサにより駆動されるプログラムで構成される。勿論、ファームウェア等で組み込むこともできる。また、運転ファイル２および部分差し替えファイル３は、メモリ上のデータを格納するためのエリアを備えたファイルである。
【０００３】現在運転中のファイル２に瑕疵がある場合、あるいは運転中ファイル２に機能を追加したい場合

には、部分差し替えファイル３を投入していく。しかし、部分差し替えファイル３を投入する前にタスク５１が被差し替え関数２１内で中断していると（中断点２１１で中断）、ローダ４により新しい差し替え関数３１，３Ｎに対してリンクが取られる。この中断タスク５１は、新しい差し替え関数３１の処理は経験していないが、差し替え処理の終了後に実行を再開すると、新しい差し替え関数３Ｎを走行することとなる。この場合、タスク５１が新しい差し替え関数３１の処理無しで関数３Ｎを走行することになるため、関数３Ｎが関数３１の処理結果を前提としているような処理を持っていれば、中断再開後、タスク５１の動作が異常になる可能性が高い。すなわち、差し替え関数３Ｎを実行するには、差し替え関数３１の処理を経由していないと矛盾が生じる論理になっている。この矛盾は、軽微な誤動作ではなく、未実装番地へアクセスしたり、データ部にアクセスしたり、あるいは誤ったアドレスのデータを破壊する等のソフトウェア障害を引き起こすため、ソフトウェアの全初期設定によりシステムの運転を回復させる必要があった。すなわち、ソフトウェア障害が発生すると、障害検出部６１により検出されてシステム全体再開処理６７が起動され、タスク等の全消去の後にシステムの運転が再開される。しかし、このような再開処理が起こると、運転中のシステムが中断し、提供中のサービスが提供できなくなるという問題がある。

【０００４】

【発明が解決しようとする課題】このように、従来のシステムにおいては、運転中ファイルを部分差し替えする場合に、差し替えプログラムの内容が正しくても、変更前に中断していたタスクが差し替え処理完了直後に処理再開した時点で処理矛盾を起こし、結局はソフトウェア障害となって、安定した運用ができなくなる場合があった。そこで、本発明の目的は、このような従来の課題を解決し、運転中の原本ファイルに対して部分差し替えファイルをロード・リンクする直前に残っていた中断タスクが、新しく盛り込まれた差し替えファイルにより未実装番地アクセス、データ部走行、データ破壊等のソフトウェア障害を引き起こす可能性があっても、ソフトウェア障害を起こしたタスクのみを消去するのみで、浮きリソースの発生を防ぎ、中断タスクでソフトウェア障害を起こしたもの以外のサービスを継続させることが可能なシステム運転中部分ファイル更新方法を提供することにある。また、本発明の他の目的は、このような部分的な再開処理で、システムの正常な運転が回復できない場合には、システムの正常な運転回復のためのガードとなるようなシステム運転中部分ファイル更新方法を提供することにある。

【０００５】

【課題を解決するための手段】上記目的を達成するため、本発明のシステム運転中部分ファイル更新方法で

は、運転中のファイルを部分的に差し替える部分ファイル更新ローダが、運転ファイルの被差し替え部分を新しい部分差し替えファイルに変更する場合に、障害検出部が差し替えファイルが組み込まれてから一定の監視時間内に発生した障害を検出し、障害分析処理部がその障害がソフトウェア要因によるものか否かを分析し、もしソフトウェア要因である場合には、タスク特定・解放処理部がその障害を発生させたプログラム並列実行処理単位を特定した後に消去し、固定点再開処理部が実行制御の特定の点から処理を再開する。さらに、障害を引き起こしたプログラム並列実行処理単位を消去した後に、後処理関数起動部が後処理プログラムを起動させて、プログラムによるリソース解放処理後に固定点再開処理部が実行制御の特定の点から処理を再開する。さらに、部分的な再開処理でシステムの正常な運転が回復できない場合には、障害の繰り返しをカウントして、しきい値を越えた場合にシステム全体を再開させることにより、システムの運転回復を保証する。

【０００６】

【発明の実施の形態】以下、本発明の実施例を、図面により詳細に説明する。図１は、本発明の一実施例を示すシステムのブロック構成図である。図１に示すように、本発明においては、従来例の図１３と同一部分を備えており、図１３と異なる部分は、再開処理部６の構成を追加した点と、部分差し替えファイル３中に後処理関数３Ｐを設けた点と、タスク５１に関連リソース５２を設けた点である。図１３と同一のものには、同一の符号を付して示している。すなわち、１はプログラムにより制御されるシステム、２はシステム建設時あるいはファイル全体更新時に設定された原本の運転ファイル、２１～２Ｎは運転ファイル２を構成する関数のうち置換される対象のもの、３は部分差し替えファイル、３１，３Ｎは部分差し替えファイルを構成する差し替え対象の関数、４は差し替え関数をメモリ上に割り付けて、原本ファイルの被差し替え対象関数の代わりにリンクし直す機能とそれをアンロード（取り外し）する機能を備えた部分ファイル変更ローダ、４１は部分更新中判定フラグ、５は実行制御部、５０１はタスク起動制御の開始点である固定点、５１は実行制御部により提供されるタスクと呼ばれる並列実行単位、５２は関連リソース、６はシステム再開処理部である。システム再開処理部６には、障害検出機構６１、ソフトウェア障害か否かを判断する分析処理部６２、システムが部分ファイル変更中か否かを確認する部分ファイル変更中判定部６３、罹障タスクを特定してこれを解放させるためのタスク特定・解放処理６４、後処理関数３Ｐを起動する後処理関数起動部６５、実行制御部５の固定点５０１に戻り、実行制御を続行する固定点再開処理６６、システムの全体的な再開を実施する全体再開処理６７、および後処理関数を起動した後も、罹障を繰り返すか否かを判定する部分再開しきい値６３

１、およびその罹障がさらに繰り返すか否かを判定するための全体再開しきい値６３２が設けられている。なお、上記障害検出機構６１のみは、ハードウェア的に割り込みを起こさせる機構であって、ハードウェア構成である。また、運転ファイル２および差し替えファイル３は、メモリのデータを格納するエリアを備えたファイルであり、それ以外の機能部は全てメモリ上に格納され、事象が発生した際にプロセッサにより駆動されるプログラムである。勿論、論理をファームウェア等で組み込んだ装置（プロセッサ）とすることも可能である。

【０００７】図１においては、運転中の原本ファイル２に対して、被差し替え対象の関数２１，２Ｎを差し替える関数３１と３Ｎに、差し替えファイル３１と３Ｎによって引き起こされ得るソフトウェア障害に関わるタスクに関連するリソースを消去するための処理を記述した関数３Ｐを添付する。さらに、再開処理部６では、障害検出部６１がローダ４によるメモリ上への割り付け後に発生した障害を検出し、分析部６２がソフトウェア障害であるか否かを判定し、部分ファイル変更中判定部６３がシステムの部分ファイル変更中であるか否かを確認し、タスク特定・解放処理防６４が罹障タスクを特定して解放し、後処理関数起動部６５が後処理関数３Ｐを起動し、全体再開処理部６７がシステムの全体的な再開を実施する。これにより、ローダ４が差し替え処理を実行する直前に中断していたタスクが、差し替えファイル３１と３Ｎがリンクされたことによって矛盾を生じ、ソフトウェア障害を引き起こした時に、それがソフトウェア障害であって、かつ罹障タスク５１を特定できた場合に、そのタスクを強制消去する。そして、そのタスクに関連していたリソースを消去するための後処理関数３Ｐを起動し、その関数に記述されている通りのリソース消去処理を行った上で実行制御５の固定点５０１に復帰する。部分ファイル差し替えの直前に中断していたタスクに関連する部分以外については、全く影響を与えることなく運転中のファイルを部分変更することが可能である。さらに、後処理関数を起動させた後に罹障が繰り返されるか否かを部分再開しきい値６３１によって判定し、もし、しきい値オーバの場合には、タスク特定・解放処理部６４によりそのタスクを強制消去する。同時に、ローダ４を起動して部分差し替えファイル３１、３Ｎをアンロードすることにより部分差し替え前の状態に戻す。また、その後も罹障を繰り返す場合には、その回数を計数して、全体再開しきい値６３２を越えた場合には、システム全体再開処理６７を起動し、システムの運転を回復させる。

【０００８】図２～図４は、本発明の動作原理を示す説明図である。なお、図１と同一のものには同一の符号を付して示している。そして、図２，図３，図４の順序で状態が変化していく様子を示している。図２は、運転中ファイルに対して部分差し替えファイルを投入する前

で、実行制御内でタスクが中断している状態を示している。すなわち、実行制御部５により制御されるタスク５１が、中断点アドレス５１１として被差し替え関数２１内の中断点２１１のアドレスを格納している。図３では、部分差し替えファイル３がシステムに割り付けられた直後にそのタスク５１が起動して、タスク５１の処理を中断点アドレス５１１から再開するが、タスク中断点５１１の前方と後方に差し替えられた関数が置かれており、そのタスク５１は差し替えられた関数の処理のうち、後ろの半分しか通らないために、差し替え関数内で処理矛盾が発生した状態を示している。すなわち、中断からの再改後、差し替え関数３Ｎを動くと、ソフトウェア異常が発生する。これにより、障害検出機構６１が起動される。また、図４では、障害検出機構６１により検出し、障害分析処理部６３で分析の結果、発生した障害がソフトウェア障害であり、罹障タスクを特定して、そのタスクのリセットとそのタスクに関係するリソースを差し替えファイルと同時に入力した後処理関数によって消去して、他の部分には影響がなく部分ファイル差し替えが完了していることを示している。

【０００９】図５～図１２は、本発明の一実施例を示す詳細動作の説明図である。前述の動作原理を、実際の動作により詳細に説明する。図５に示すように、部分差し替えファイル３が割り付けられる前で、被差し替え関数２１の中でタスク５１が中断しているものとする。タスク５１の実行は実行制御部５により制御されるが、タスク５１は中断点アドレス５１１として、被差し替え関数２１内の中断点２１１のアドレスを格納している。この時、図６に示すように、ローダ４により差し替えファイル３内の関数３１，３Ｎが割り付けられ、被差し替え関数２１，２Ｎとの間でリンケージがなされる（リンケージ用ジャンプ３１１，３１Ｎ）。このようにして、部分差し替えファイル３が投入されると、ローダ４の機能により差し替え関数３１と３Ｎ、および後処理関数３Ｐがシステム上に割り付けられる。なお、リンケージ処理直前に中断していたタスク５１は、リンケージ後までそのまま残る。次に、図７に示すように、ローダ４は、部分差し替え関数３１と３Ｎが割り付けを終了すると、実行制御部５に制御を渡し、実行制御部５はタスク５１を中断点から起動させて中断点アドレス２１１から処理を再開させようとする。しかし、タスク５１は該中断点にて就寝していたため、差し替え関数３１の処理を経験させずに差し替え関数３Ｎの処理を走行しようとする。そのため、本来、関数３１と３Ｎの組み合わせにおいて、１つの正当な処理がなされるところが実際にはそのようにならないために、処理矛盾が発生して、システム障害となる（障害を×で示す）。

【００１０】次に、図８においては、障害検出機構６１が障害を検出し、障害分析処理部６２でソフトウェア要因の障害であることを判定すると、部分更新中判定部６

３により部分ファイル更新ローダ４内の部分更新中フラグ４１を判定し、かつ部分再開しきい値６３１を判定する。部分更新中でしきい値以内であれば、タスク特定解放処理部６４において、罹障を引き起こしたタスクを特定し、そのタスク５１を消去する（消去を×で示す）。タスク消去に引き続いて、図９に示すように、後処理関数起動部６５により後処理関数３Ｐを起動し、そのタスクに関連していたリソース５１１を消去する（消去を×で示す）。後処理関数３Ｐは、差し替え関数３１，３Ｎをコーディングするプログラマが、中断タスクが存在して引き起こす可能性のある障害時に関連リソースを特定して解放する処理を記述したものである。この関数は、この時のみ起動されるものであるが、その起動の仕方は実装に依存する。本実施例では、関数名に特定の名称またはサフィックスを付加してその関数を識別し、割り付け位置はその名称をアドレスを対応させるデータとして所持しているものとする。なお、図８および図９において、実装によっては、他の処理が割り込まないように、実行制御部５に対して割り込みマスクを設定する。

【００１１】図１０は、関連リソース消去の後に、固定点再開処理部６６により実行制御部５の固定点５０１から通常の運転を再開させる流れを示したものである。すなわち、関連リソース５２の消去から戻ると、固定点再開処理部６６を経由して、実行制御部５の固定点５０１からシステムの運転を再開させる。図１１は、更新部分しきい値をオーバした場合の処理の説明図である。後処理関数起動部６５を起動してタスクやリソースを消去しても、障害を繰り返して、部分再開しきい値６３１をオーバした場合には、新たに割り付けた部分差し替えファイル３１，３Ｎに瑕疵があり、タスクの強制終了では罹障を回復できない場合であるとみなす。この場合には、タスクを解放するとともに、部分再開ローダ４（部分ファイル変更ローダと同じ）により部分差し替えファイル３１，３Ｎの割り付けを元に戻し（アンロードする）、リンケージを戻して固定点再開処理部６６により実行制御部５の固定点５０１から通常の運転を再開させる処理を行う。なお、上記しきい値は、一定の条件、例えば一定時間という条件でリセットし、無関係な要因発生によりしきい値オーバする確率を下げている。図１２は、部分差し替えファイルを戻しても回復できないような障害に陥った場合、および今回の部分ファイル差し替えと関係ない要因でソフトウェア障害が起こっている場合か否かを判断する動作を示している。例えば、一定時間内にソフトウェア障害が全体再開しきい値６３２をオーバしたならば、全体再開処理６７によりシステム全体再開を行うことにより、システムを回復できる確率を高めることができる。

【００１２】

【発明の効果】以上説明したように、本発明によれば、プログラムにより制御されるシステムで、運転ファイルを部分的に変更する際に、複数箇所の関数を修正しても中断タスクがあるために矛盾が発生してシステム障害になるところ、矛盾が発生したタスクをリセットするとともに、そのタスクに関するリソースを特定して初期設定する関数を起動することにより、部分的な影響のみでシステム全体の運転の妨げをなくすことができる。また、上記の処理を行っても罹障を繰り返す場合には、部分差し替えファイルを戻して固定点再開させるか、あるいはシステム全体を再開させることにより、システムの正常な運転回復のためのガードを行うことができる。その結果、本発明によれば、運転中のファイルを部分更新する場合のシステム障害を防止することができるので、安定してシステムを運転させることができる。

【図面の簡単な説明】

【図１】本発明によるシステム運転中部分ファイル更新の動作原理を示す図である。

【図２】本発明によるシステム運転中部分ファイル更新の概略動作の説明図（部分差し替えファイル投入前）である。

【図３】同じく概略動作の説明図（部分差し替え関数内で処理矛盾が発生した状態）である。

【図４】同じく概略動作の説明図（罹障タスクのリセットと、リソースを消去する処理）である。

【図５】本発明の一実施例を示すシステム運転中部分ファイル更新の詳細動作の説明図（部分差し替えファイルの割り付け前）である。

【図６】同じく詳細動作の説明図（差し替え関数および後処理関数の割り付け処理）である。

【図７】同じく詳細動作の説明図（処理矛盾の発生とシステム障害）である。

【図８】同じく詳細動作の説明図（部分更新中でしきい値以内のため、タスクの消去を行う）である。

【図９】同じく詳細動作の説明図（関連リソースの消去）である。

【図１０】同じく詳細動作の説明図（固定点再開処理）である。

【図１１】同じく詳細動作の説明図（部分しきい値をオーバした場合）である。

【図１２】同じく詳細動作の説明図（部分差し替えファイルを戻しても回復できない障害の場合）である。

【図１３】従来のシステム運転中部分ファイル更新の動作説明図である。

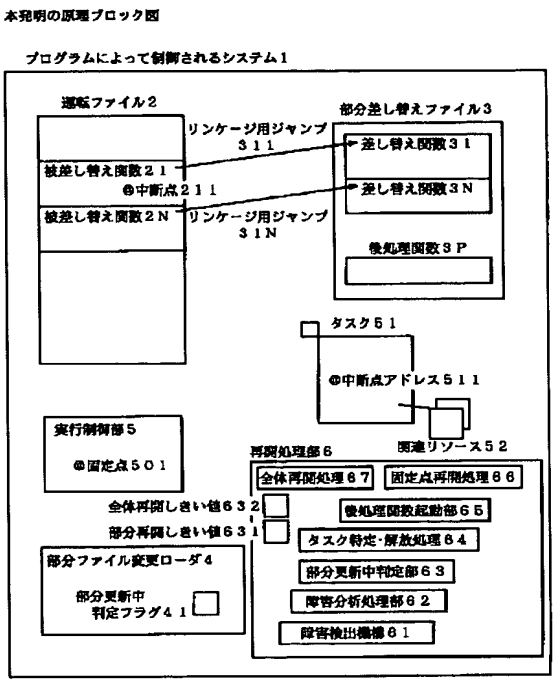
【符号の説明】

１…プログラムにより制御されるシステム、２…運転ファイル、２１～２Ｎ…運転ファイルを構成するプログラム単位（関数）（差し替えられる対象の被差し替え関数）、２１１…タスク中断点、３…部分差し替えファイル、４…部分ファイル変更ローダ、３Ｐ…差し替え関数を入れたことにより発生し得るソフトウェア障害に関連するリソースを消去する後処理を記述した後処理関数、

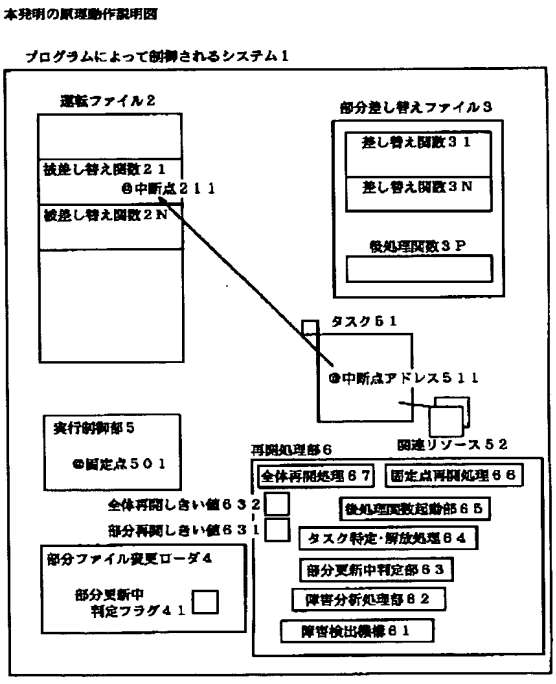
３１～３Ｎ…差し替えファイル内で、差し替え用の部分差し替え関数、４１…部分更新中判定フラグ、５…タスクの実行を司る実行制御部、５１…プログラム実行単位タスク、５０１…実行制御の起点となる固定点、５１１…タスク５１が関数中で中断しているアドレスを示す中断点アドレス、５２…タスクが関連するリソース、６１…障害検出機構、６…障害発生した場合の回復処理を司る再開処理部、６２…検出した障害がソフトウェア障害か否かを判定する障害分析処理部、６３…障害発生時、部分更新中フラグ４１をチェックして部分更新中か否かを判定する部分更新中判定部、６３１…障害が繰り返し、部分差し替えファイルをアンロードするか否かを判定する部分再開しきい値、６３２…障害が繰り返し、システム全体再開を起動するか否かを判定する全体再開しきい値、６４…罹障タスクを特定し､解放するタスク特定・解放処理部、

６５…差し替えファイル３の投入と同時に割り付けられる後処理関数３Ｐを起動する後処理関数起動部、６６…実行制御の固定点から再開するための処理を行う固定点再開処理部、６７…システム全体の再開を処理する全体再開処理部。

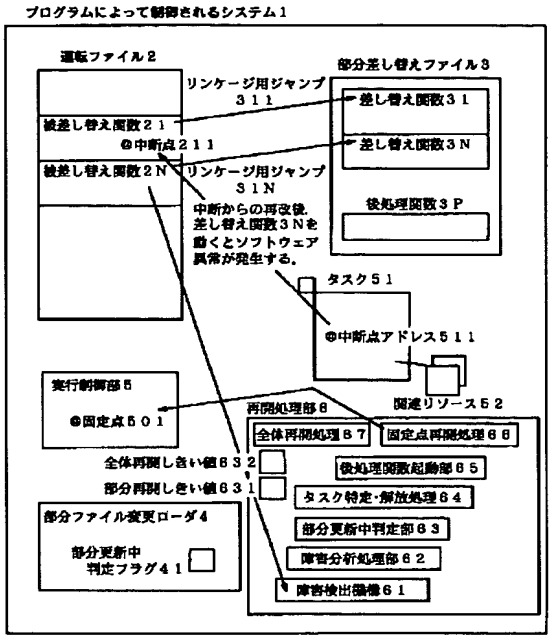
【図１】



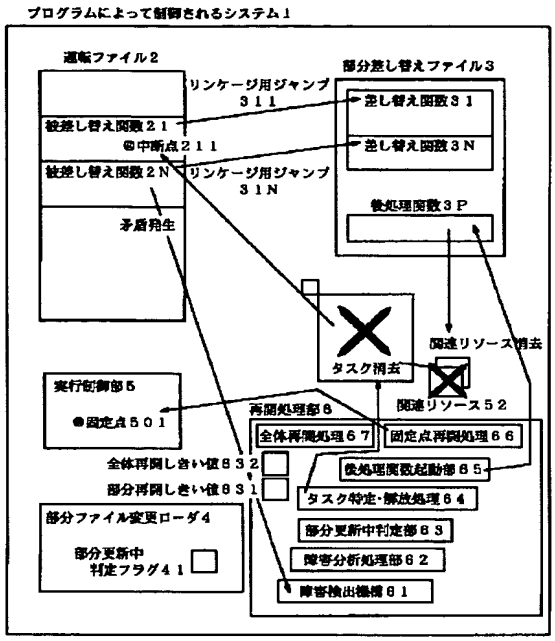
【図２】



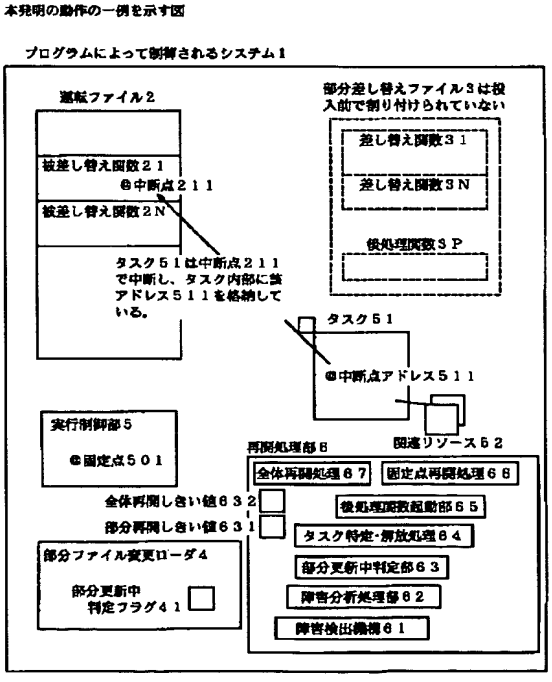
【図３】


プログラムによって制御されるシステム１
運転ファイル２
部分差し替えファイル３
リンケージ用ジャンプ３１１
被差し替え関数２１
差し替え関数３１
◎中断点２１１
差し替え関数３Ｎ
被差し替え関数２Ｎ
リンケージ用ジャンプ３１Ｎ
後処理関数３Ｐ
中断からの再改後、差し替え関数３Ｎを動くとソフトウェア異常が発生する。
タスク５１
◎中断点アドレス５１１
実行制御部５
◎固定点５０１
再開処理部６
関連リソース５２
全体再開処理６７
固定点再開処理６６
全体再開しきい値６３２
後処理関数起動部６５
部分再開しきい値６３１
タスク特定・解放処理６４
部分ファイル変更ローダ４
部分更新中判定部６３
部分更新中判定フラグ４１
障害分析処理部６２
障害検出機構６１

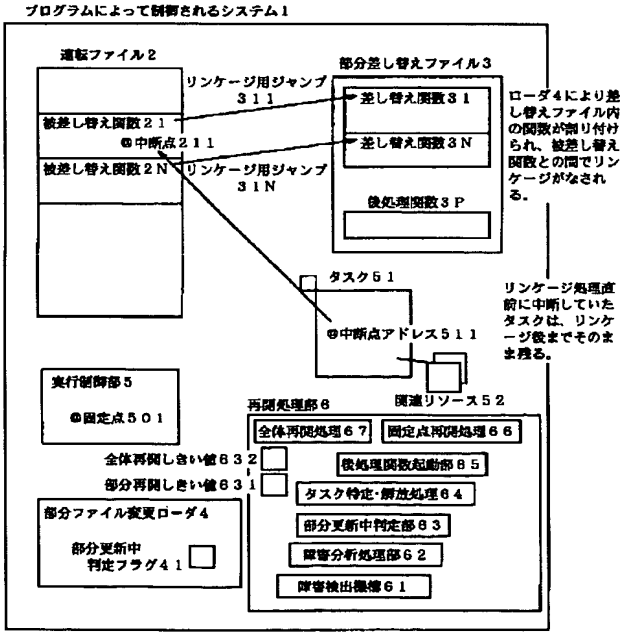
【図４】


プログラムによって制御されるシステム１
運転ファイル２
部分差し替えファイル３
リンケージ用ジャンプ３１１
被差し替え関数２１
差し替え関数３１
◎中断点２１１
差し替え関数３Ｎ
被差し替え関数２Ｎ
リンケージ用ジャンプ３１Ｎ
後処理関数３Ｐ
矛盾発生
タスク消去
関連リソース消去
実行制御部５
◎固定点５０１
再開処理部６
関連リソース５２
全体再開処理６７
固定点再開処理６６
全体再開しきい値６３２
後処理関数起動部６５
部分再開しきい値６３１
タスク特定・解放処理６４
部分ファイル変更ローダ４
部分更新中判定部６３
部分更新中判定フラグ４１
障害分析処理部６２
障害検出機構６１

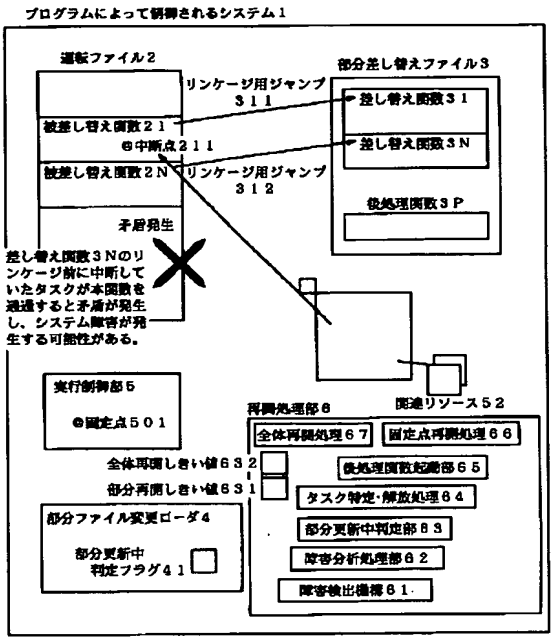
【図５】


本発明の動作の一例を示す図
プログラムによって制御されるシステム１
運転ファイル２
部分差し替えファイル３は投入前で割り付けられていない
差し替え関数３１
被差し替え関数２１
◎中断点２１１
差し替え関数３Ｎ
被差し替え関数２Ｎ
後処理関数３Ｐ
タスク５１は中断点２１１で中断し、タスク内部に飛アドレス５１１を格納している。
タスク５１
◎中断点アドレス５１１
実行制御部５
◎固定点５０１
再開処理部６
関連リソース５２
全体再開処理６７
固定点再開処理６６
全体再開しきい値６３２
後処理関数起動部６５
部分再開しきい値６３１
タスク特定・解放処理６４
部分ファイル変更ローダ４
部分更新中判定部６３
部分更新中判定フラグ４１
障害分析処理部６２
障害検出機構６１

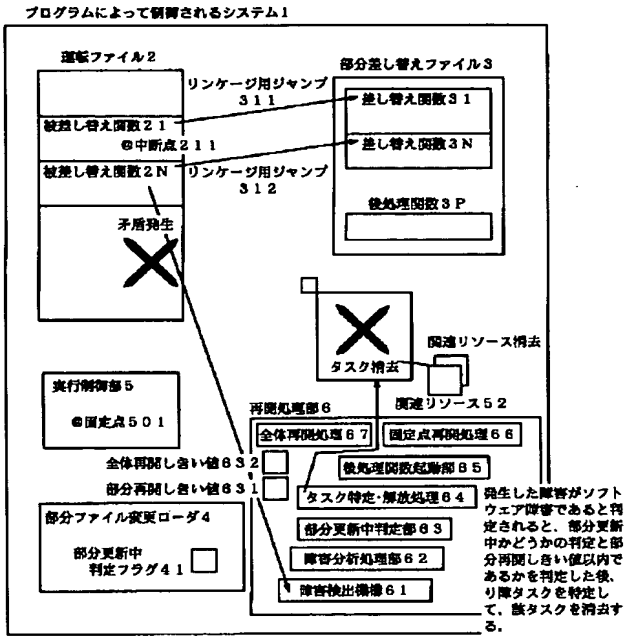
【図６】


プログラムによって制御されるシステム１
運転ファイル２
部分差し替えファイル３
リンケージ用ジャンプ３１１
被差し替え関数２１
差し替え関数３１
◎中断点２１１
差し替え関数３Ｎ
被差し替え関数２Ｎ
リンケージ用ジャンプ３１Ｎ
後処理関数３Ｐ
ローダ４により差し替えファイル内の関数が割り付けられ、被差し替え関数との間でリンケージがなされる。
タスク５１
◎中断点アドレス５１１
リンケージ処理直前に中断していたタスクは、リンケージ後までそのまま残る。
実行制御部５
◎固定点５０１
再開処理部６
関連リソース５２
全体再開処理６７
固定点再開処理６６
全体再開しきい値６３２
後処理関数起動部６５
部分再開しきい値６３１
タスク特定・解放処理６４
部分ファイル変更ローダ４
部分更新中判定部６３
部分更新中判定フラグ４１
障害分析処理部６２
障害検出機構６１

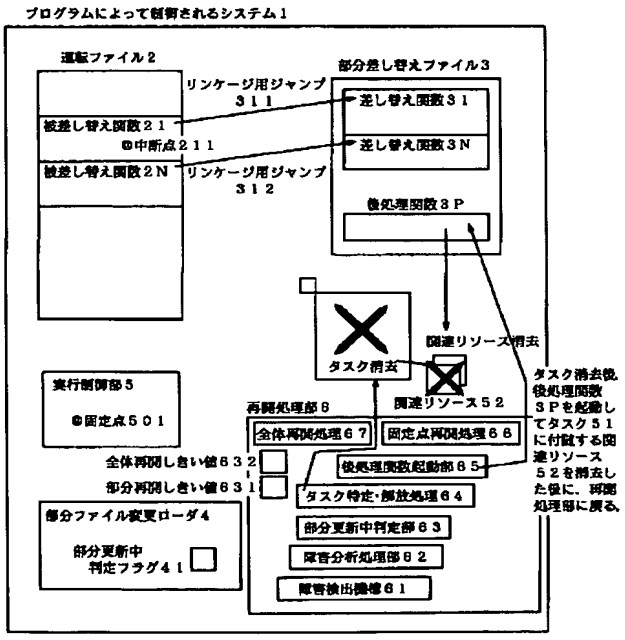
【図７】


プログラムによって制御されるシステム１
運転ファイル２
部分差し替えファイル３
リンケージ用ジャンプ３１１
差し替え関数３１
被差し替え関数２１
◎中断点２１１
差し替え関数３Ｎ
被差し替え関数２Ｎ
リンケージ用ジャンプ３１２
後処理関数３Ｐ
矛盾発生
差し替え関数３Ｎのリンケージ前に中断していたタスクが本関数を通過すると矛盾が発生し、システム障害が発生する可能性がある。
実行制御部５
◎固定点５０１
再開処理部６
関連リソース５２
全体再開処理６７
固定点再開処理６６
全体再開しきい値６３２
後処理関数起動部６５
部分再開しきい値６３１
タスク特定・解放処理６４
部分ファイル変更ローダ４
部分更新中判定部６３
部分更新中判定フラグ４１
障害分析処理部６２
障害検出機構６１

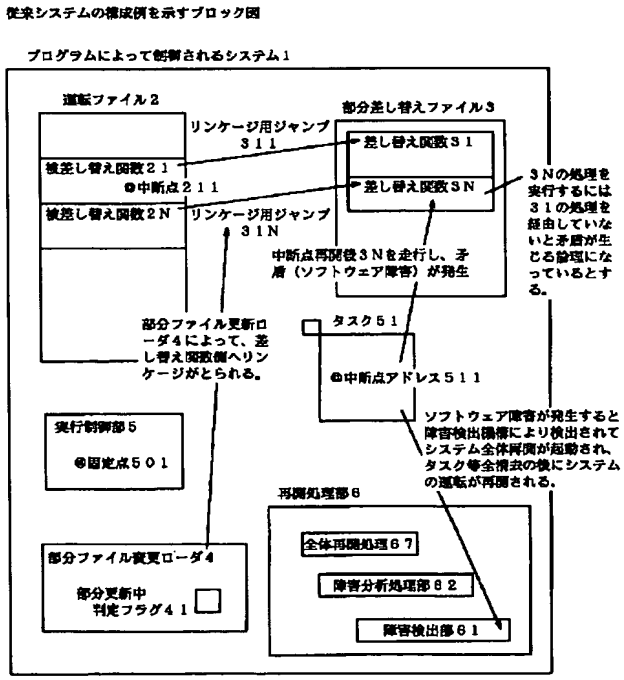
【図８】


プログラムによって制御されるシステム１
運転ファイル２
部分差し替えファイル３
リンケージ用ジャンプ３１１
差し替え関数３１
被差し替え関数２１
◎中断点２１１
差し替え関数３Ｎ
被差し替え関数２Ｎ
リンケージ用ジャンプ３１２
後処理関数３Ｐ
矛盾発生
タスク消去
関連リソース消去
実行制御部５
◎固定点５０１
再開処理部６
関連リソース５２
全体再開処理６７
固定点再開処理６６
全体再開しきい値６３２
後処理関数起動部６５
部分再開しきい値６３１
タスク特定・解放処理６４
部分ファイル変更ローダ４
部分更新中判定部６３
部分更新中判定フラグ４１
障害分析処理部６２
障害検出機構６１
発生した障害がソフトウェア障害であると判定されると、部分更新中かどうかの判定と部分再開しきい値以内であるかを判定した後、り障タスクを特定して、該タスクを消去する。

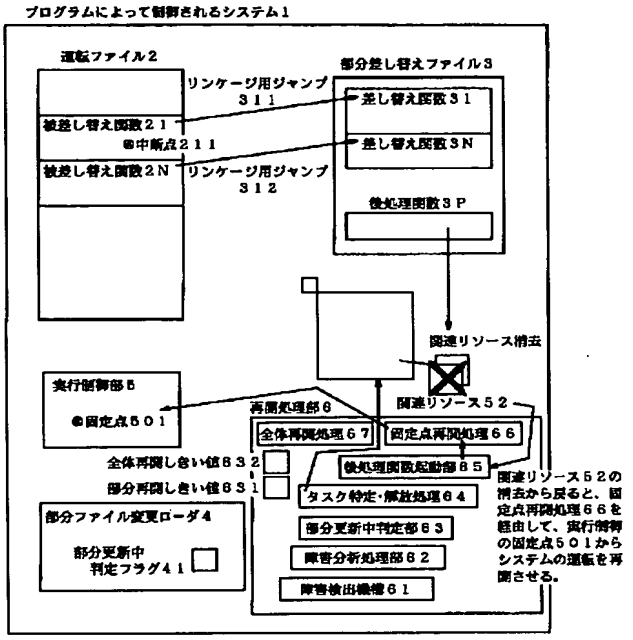
【図９】


プログラムによって制御されるシステム１
運転ファイル２
部分差し替えファイル３
リンケージ用ジャンプ３１１
差し替え関数３１
被差し替え関数２１
◎中断点２１１
差し替え関数３Ｎ
被差し替え関数２Ｎ
リンケージ用ジャンプ３１２
後処理関数３Ｐ
タスク消去
関連リソース消去
実行制御部５
◎固定点５０１
再開処理部６
関連リソース５２
全体再開処理６７
固定点再開処理６６
全体再開しきい値６３２
後処理関数起動部６５
部分再開しきい値６３１
タスク特定・解放処理６４
部分ファイル変更ローダ４
部分更新中判定部６３
部分更新中判定フラグ４１
障害分析処理部６２
障害検出機構６１
タスク消去後、後処理関数３Ｐを起動してタスク５１に付随する関連リソース５２を消去した後に、再開処理部に戻る。

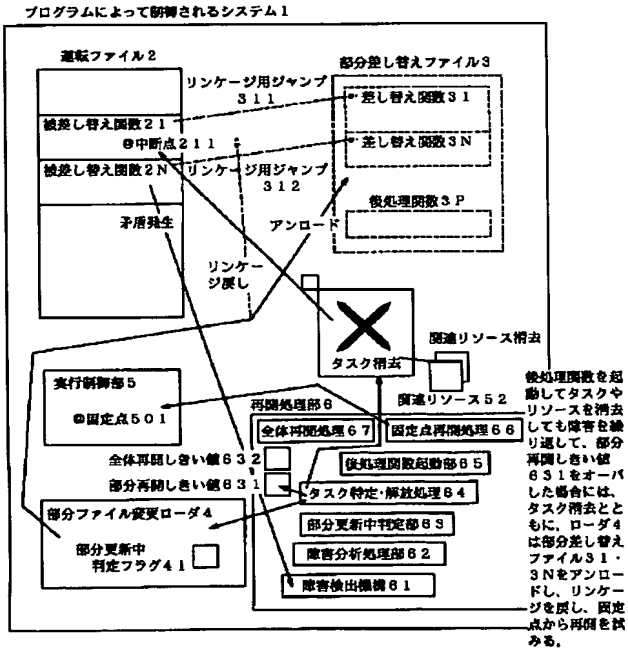
【図１３】


従来システムの構成例を示すブロック図
プログラムによって制御されるシステム１
運転ファイル２
部分差し替えファイル３
リンケージ用ジャンプ３１１
差し替え関数３１
被差し替え関数２１
◎中断点２１１
差し替え関数３Ｎ
被差し替え関数２Ｎ
リンケージ用ジャンプ３１Ｎ
３Ｎの処理を実行するには３１の処理を経由していないと矛盾が生じる論理になっているとする。
中断点再開後３Ｎを走行し、矛盾（ソフトウェア障害）が発生
部分ファイル更新ローダ４によって、差し替え関数側へリンケージがとられる。
タスク５１
◎中断点アドレス５１１
実行制御部５
◎固定点５０１
ソフトウェア障害が発生すると障害検出機構により検出されてシステム全体再開が起動され、タスク等全消去の後にシステムの運転が再開される。
再開処理部６
部分ファイル変更ローダ４
部分更新中判定フラグ４１
全体再開処理６７
障害分析処理部６２
障害検出部６１

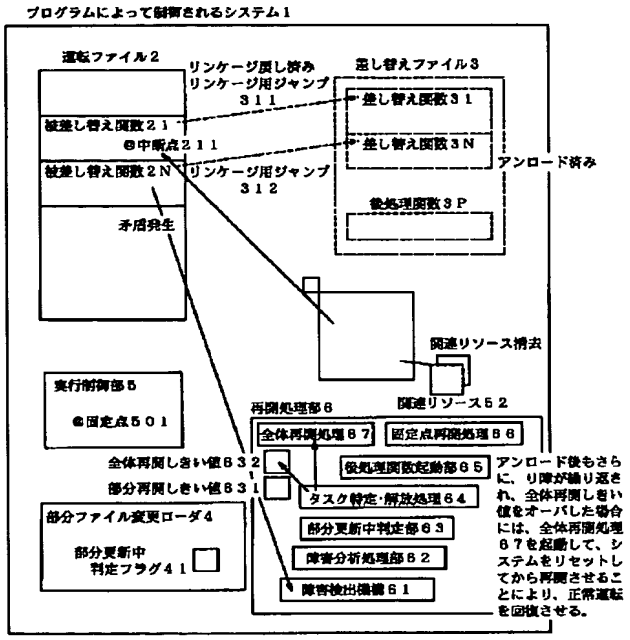
【図１０】


プログラムによって制御されるシステム１
運転ファイル２
部分差し替えファイル３
リンケージ用ジャンプ３１１
被差し替え関数２１
差し替え関数３１
●中断点２１１
被差し替え関数２Ｎ
差し替え関数３Ｎ
リンケージ用ジャンプ３１２
後処理関数３Ｐ
関連リソース消去
実行制御部５
●固定点５０１
再開処理部６
関連リソース５２
全体再開処理６７
固定点再開処理６６
全体再開しきい値６３２
後処理関数起動部６５
部分再開しきい値６３１
タスク特定・解放処理６４
部分ファイル変更ローダ４
部分更新中判定部６３
部分更新中判定フラグ４１
障害分析処理部６２
障害検出機構６１
関連リソース５２の消去から戻ると、固定点再開処理６６を経由して、実行制御の固定点５０１からシステムの運転を再開させる。

【図１１】


プログラムによって制御されるシステム１
運転ファイル２
部分差し替えファイル３
リンケージ用ジャンプ３１１
被差し替え関数２１
差し替え関数３１
●中断点２１１
被差し替え関数２Ｎ
リンケージ用ジャンプ３１２
差し替え関数３Ｎ
後処理関数３Ｐ
予盾発生
アンロード
リンケージ戻し
タスク消去
関連リソース消去
実行制御部５
●固定点５０１
再開処理部６
関連リソース５２
全体再開処理６７
固定点再開処理６６
全体再開しきい値６３２
後処理関数起動部６５
部分再開しきい値６３１
タスク特定・解放処理６４
部分ファイル変更ローダ４
部分更新中判定部６３
部分更新中判定フラグ４１
障害分析処理部６２
障害検出機構６１
後処理関数を起動してタスクやリソースを消去しても障害を繰り返して、部分再開しきい値６３１をオーバした場合には、タスク消去とともに、ローダ４は部分差し替えファイル３１・３Ｎをアンロードし、リンケージを戻し、固定点から再開を試みる。

【図１２】



フロントページの続き

(72)発明者　山田　哲靖
東京都新宿区西新宿三丁目19番２号　日本電信電話株式会社内

(72)発明者　越智　憲一
東京都港区芝五丁目７番１号　日本電気株式会社内